Diploma project

Programming and analysing measurement experiments using the PicoCricket system

Sergi Erena Estrada

Computer Science speciality in electronic systems

Project coordinator: Aleksander Zaigrajew

Guarantor: *Piotr* Bała

Toruń, June of 2012

Table of contents:

Introduction to PicoCricket

There's thousands ways to program microcontrollers, from using assembler to program in a really low level until using C or other high-level languages. The low-level languages require a good knowledge of hardware and electronics because it is directly related with positions of memory, hardware operators, instructions, voltages, overflows and so on. However the high-level languages have plenty of tools to make programming more comfortable and clearing work to the programmer therefore the results are more attractive and the code is clearer than assembler.

In the realization of this project I worked using a very high-level language, indeed there's no language, just blocks to play with. Each block represents an instruction such as mathematical operators, iterators, conditionals, flow instructions, logical operators, memory spaces, input operators and even custom procedures.

HARDWARE:

Aside from these blocks there are also blocks to get and output data. These blocks can be grouped in 2 categories:

1 **Sensors**: Used to get information from outside. There are 4 kinds of data get:
   - Intensity of Light
   - Intensity of Sound
   - Electrical resistance information
   - Pressed button as boolean information

2 **Actuators**: Used to output information. There are 4 kinds to output data:
   - Light
   - Playing sounds
   - Motor spinning
   - Displaying 3 digit numbers

3 **Other equipment**: Inside the PicoCricket box there are other equipment:
   - Beamer and PicoCricket computer

- USB Cable
- LEGO pieces and other decorative staff

SOFTWARE:

All these blocks need to be programmed and the editor/compiler available in the PicoCricket box is the "PicoBlocks editor". This editor has a friendly interface:

The PicoBlocks interface:

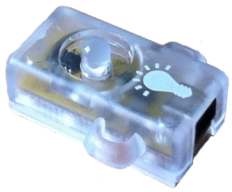
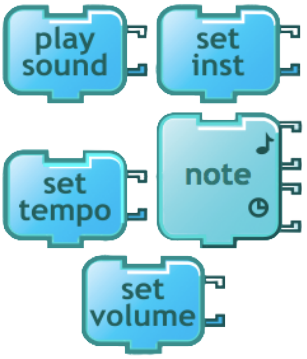
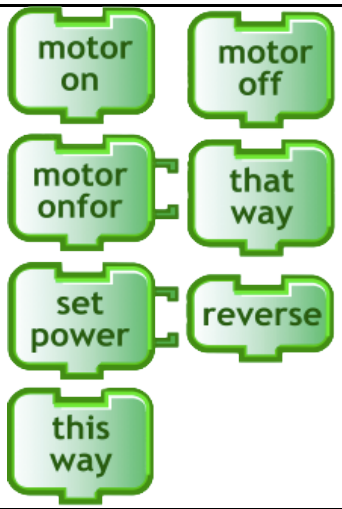
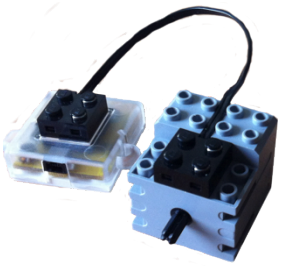
1) CATEGORY SELECTION: Here is where we can open the different categories to see what elements we have. There are 8 different kinds:

- **Lights**: Actions to use with LEDs and Display
- **Sound**: Actions to make sound signals with the PicoCricket and with the speaker
- **Action**: Set of actions to use with the Motor
- **Sensors**: A large set of blocks to get data from sensors

- **Data**: Useful blocks to store data from sensors. There's also the possibility to draw plots with the stored data.
- **Flow**: Set of structure blocks to make sequences, conditions and to define the flow for the algorithm
- **Numbers**: Operations to make mathematical calculations
- **MyBlocks**: There are customizable blocks and memory spaces to store/write/read data

2) ELEMENT SELECTION: In this second section, once we have chosen the category, we will have the elements shown in that column. There are two main categories:

SENSORS			
Light Sensor	Gets the brightness of light. The more intensity the light has, the higher is the value reported.		
Sound Sensor	Reports the volume of sound.		
Touch Sensor	Reports if the button is pressed. The report is a boolean value.		
Resistance Sensor	It gets the resistance in the circuit formed by the alligator-clip cables. Different materials have different resistances and this sensor can measure them. There's also the possibility to use the sensor as a boolean detector.		

ACTUATORS			
Light	This is a LED, which can be turned on/off from the algorithm but also there's the possibility to change the colour and the intensity.		
Sound box	This sound box is a speaker with some default sounds to play. There's also the possibility to create melodies and play instruments.		
Motor and Motor board	This motor generates a circular movement. There's 3 parameters to set: ON/OFF, power and direction		
Display	This display has 3 digits to show a range number from 000 to 999		

3) WORKSPACE: In this section is where we drag&drop blocks and other elements to join them together in order to create functional algorithms. Here is where we build the algorithm using what we have. There's also an option in the workspace to see the same algorithm but instead of blocks, we can see it in words and written instructions.

4) TOOLS: In this section there is a set of useful tools to select, cut, copy and paste blocks. There are:

- **Arrow:** Is used to select blocks and also to drag and drop entire stacks of blocks.
- **Magic wand:** Is used to compile and run the program
- **Scissors:** Cut function, is the same as Ctrl+X in Windows or ⌘+X in OSX.
- **Stamper:** As the name says is to paste whatever we cut before
- **Help:** This useful tool allows us to know all information about each element in PicoBlocks.
- **Wand 2:** This wand is used to compile and run a block in the background at the same time we are running another stack in the foreground.
- **Tags:** This useful function allows us to rename an element in case of having 2 or more of the same type. So then we can call just the element we want between different elements of the same kind.
- **Do and Undo:** This function allows us to rectify a movement we did before.
- **STOP:** This button breaks off the running process.

Objectives

The aim of this project is to get used to another kind of programming. Since now, I used very complex programming languages to develop applications or even to program microcontrollers, but PicoCricket system is the evidence that we don't need so complex development tools to get functional devices.
PicoCricket system is the clear example of simple programming to make devices work the way we programmed it. There's an easy but effective way to program small devices just saying what we want them to do. We cannot do complex algorithms and mathematical operations but we can program them in a short time. Nowadays, the easier and faster we produce, the more we earn. So the tendency is to develop fast, cheap and easy, and PicoCricket system can do it.

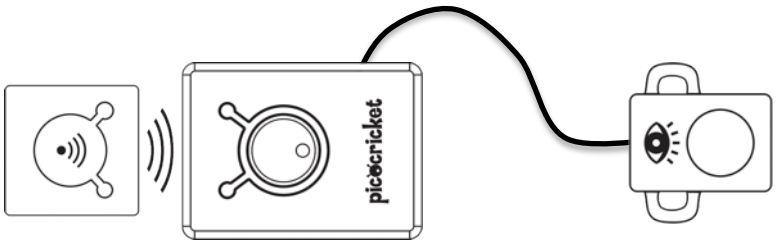
Light detector experiments

The light detector can measure the intensity of the incoming light. The intensity of the light is given in a value range between 0 and 100, where 0 means no light and 100 means the highest light.

1) Getting the light intensity

This experiment consist on measuring the intensity of the ambient light during 5 seconds, making changes on the light to see changes in the plot.

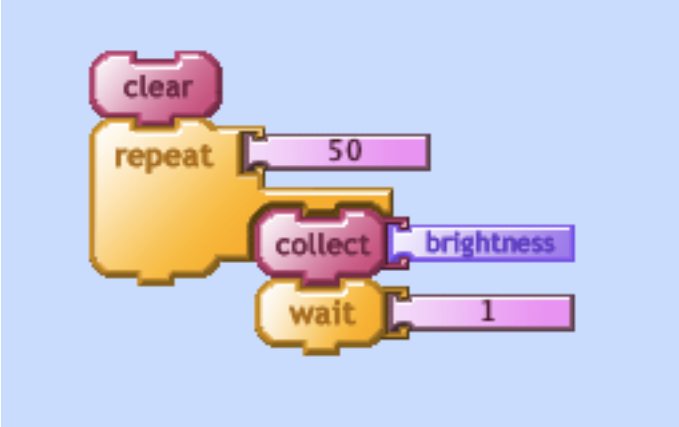
Schema



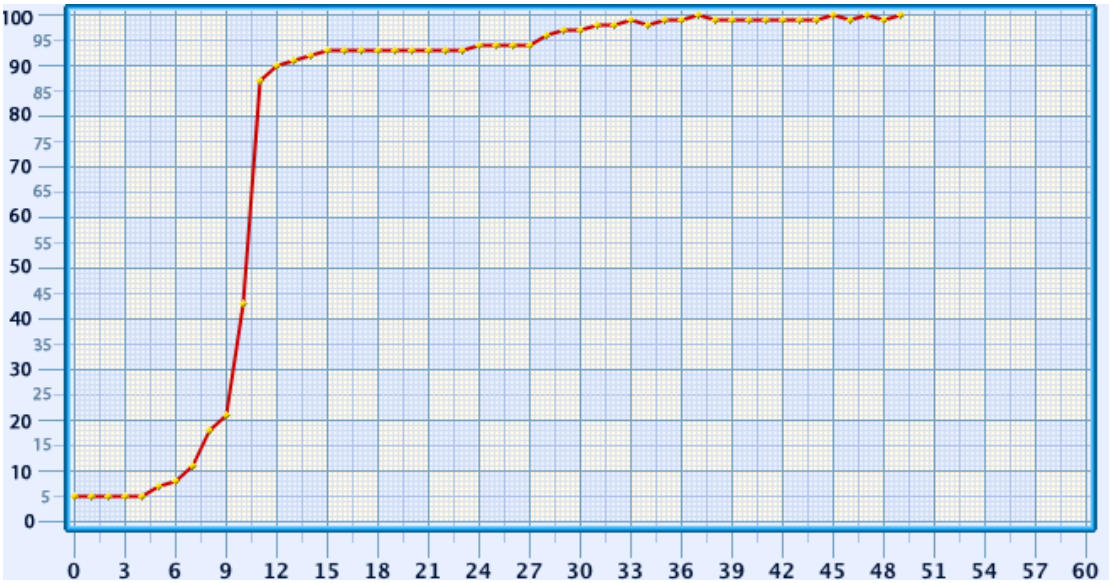
I only needed the light detector to do this experiment. This is a simple experiment to get light data.

Algorithm

The algorism used to do this experiment is simply a loop that gets the light intensity value every 10ms using the operation **collect**. Before entering into the loop we just clear the memory due to write the new incoming data, and into the loop we just get the value and after that we just wait for 1 tick (10ms).



It consists on plotting the brightness (or light intensity) every 10ms for 50 times, so each 10 times is the measurement of one second. The output graph is the following:

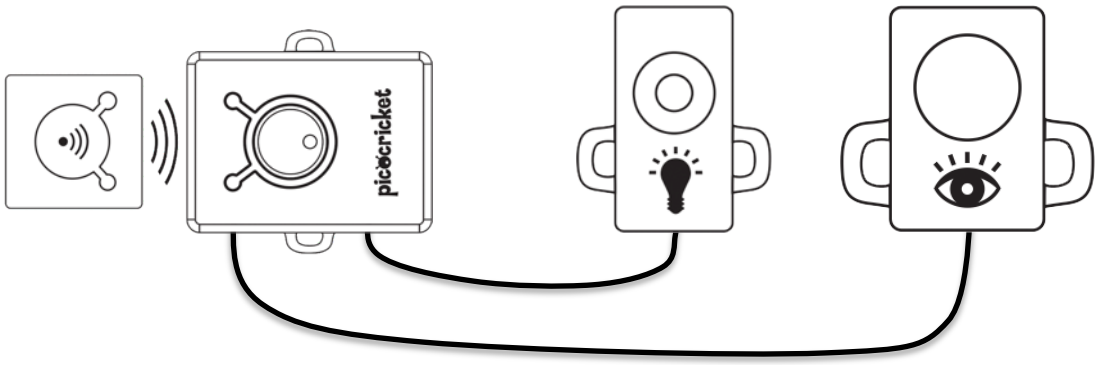


In that XY graph, X is each measurement (in my algorithm is taken each 10ms) and Y is the intensity of the light. Another output format is a list of all measurements in a .txt file that can be used as input data for other software, which needs this data.

2) Turning on the LED in function of the ambient light

The objective of this experiment is to simulate an automatic switch for a light depending on the natural light. On one hand when there's few ambient light, the LED turns brighter to illuminate the room and on the other hand when the room becomes lighter the LED illuminate less.

Schema



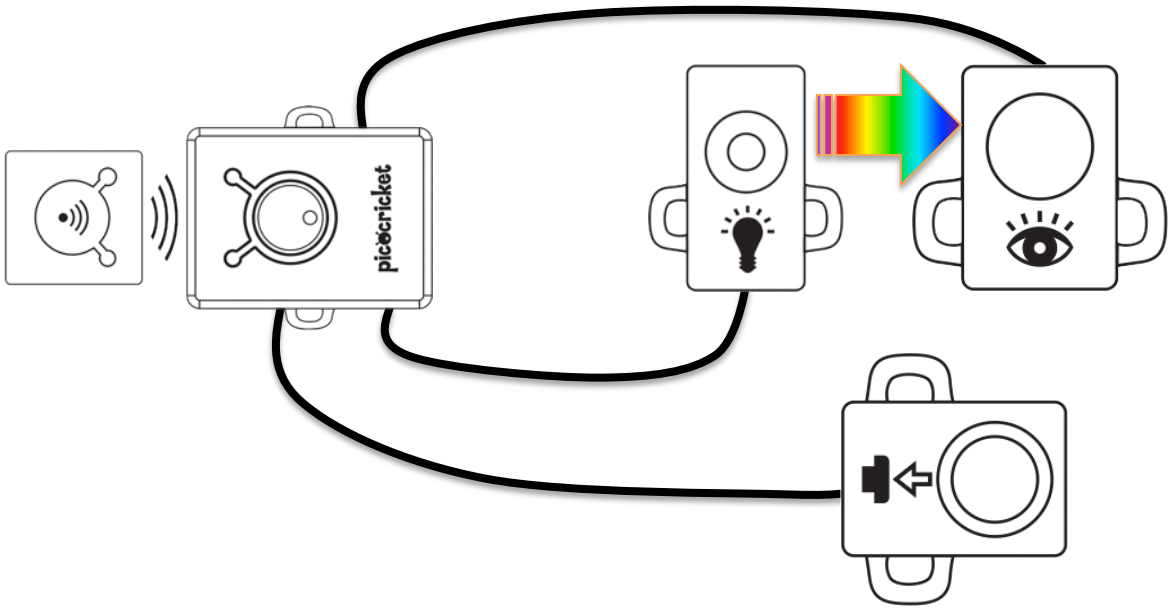
Algorithm

The algorithm used for this experiment is a simple infinite loop that bright the LED with the inversely proportional bright detected in the input. Into the loop we select a color and then we turn it on with the inverted value of the incomming brightnes. As the value must be between 0 and 100, we set the output to100 minus the brightness.

3) Getting the intensity of different light frequencies

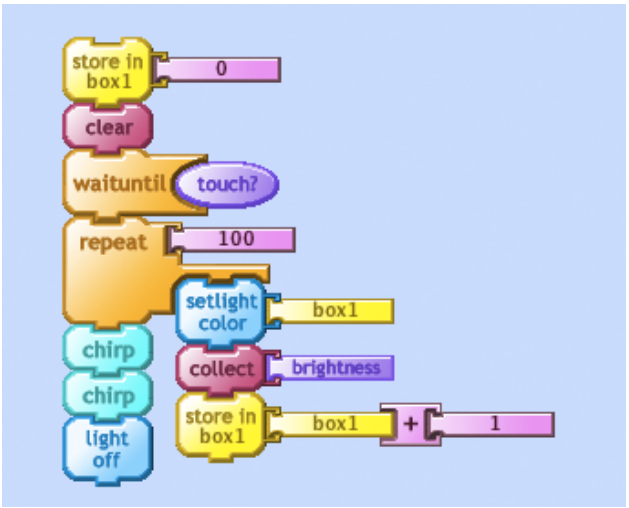
The objective of this experiment is to get the frequency of an incoming light using the light sensor, therefore I put the light detector and the LED facing each other. It is known that every colour has each frequency of light, but we want to verify if the intensity of light is the same changes for different light frequencies.

Schema

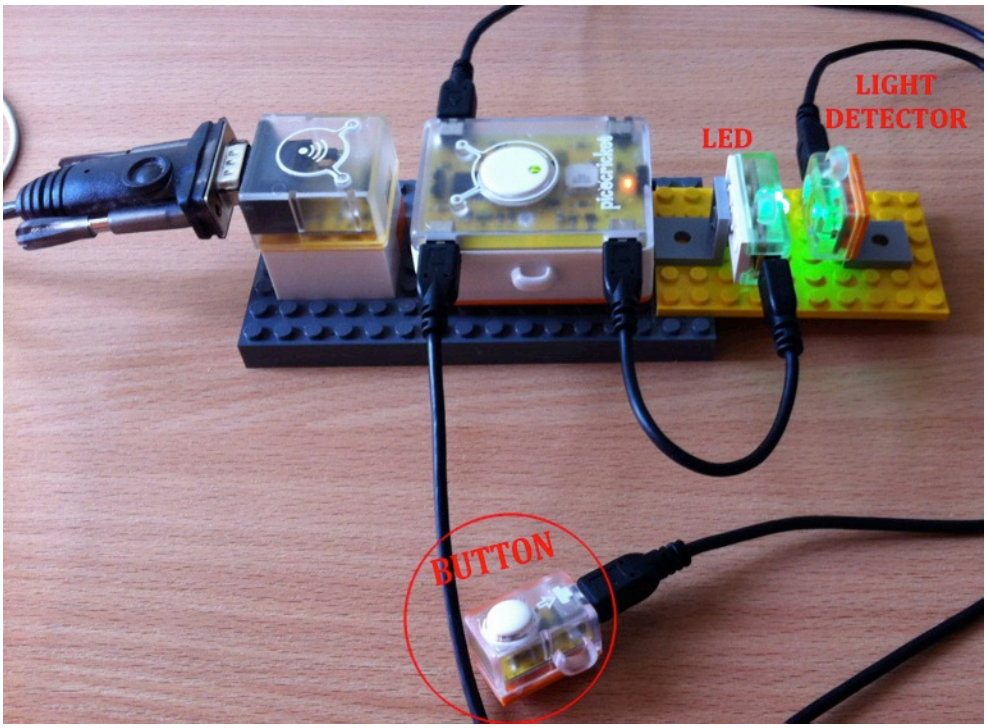


Algorithm

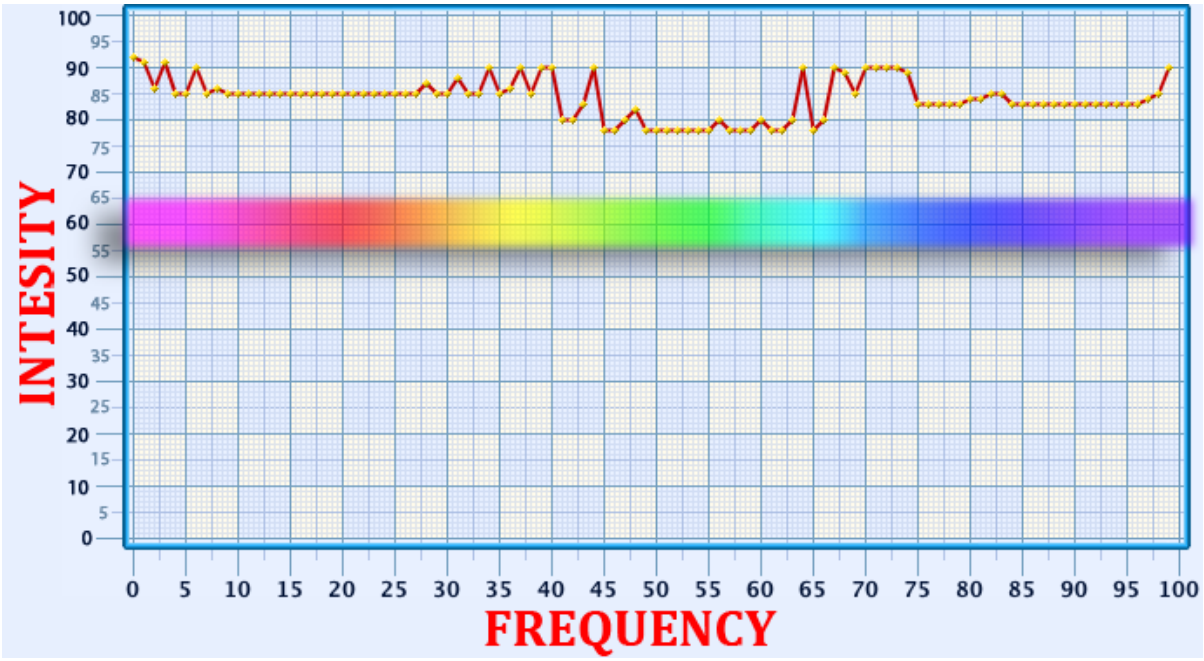
We need to get the light intensity for all range of frequencies (visible light frequencies), therefore the algorithm must bright the LED with all range of colours. The PicoCricket system LED can show colours into a range of 100 (starting with 0-purple to 100-blue), so the LED must show all colours and the light sensor must plot the intensity of each one. Into the loop, first we set the colour to **box1** value, which is the number of loop, then we get the value of brightness and the **box1** value is increased:



Once compiled the scan of the intensity of all frequencies will start when the button is pressed.



The output plot for the test is the following:



Conclusion

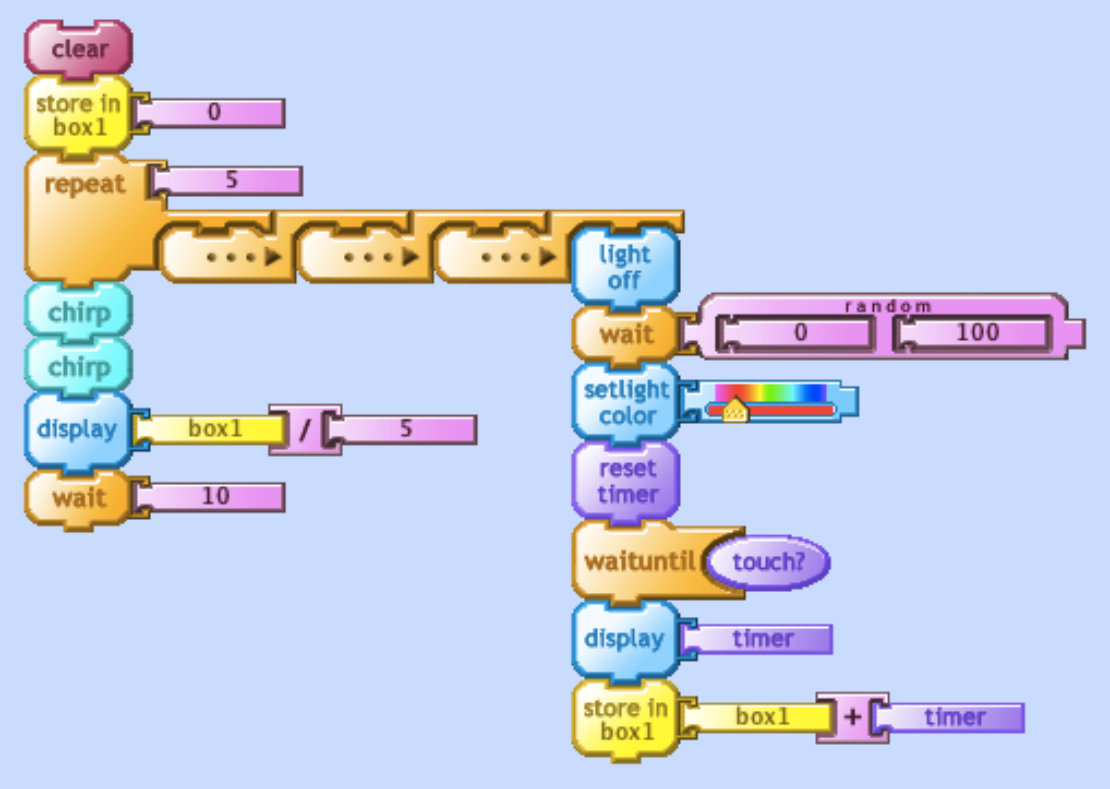
We can consider that the light with different frequencies has different intensities for each kind of colour (frequency). The frequency with less intensity of light is the green light and the frequency more intensity is the purple, the yellow and the blue ones.

We can conclude that the energy that LED emits depends on certain part on the frequency used.

4) Reflex experiment with light

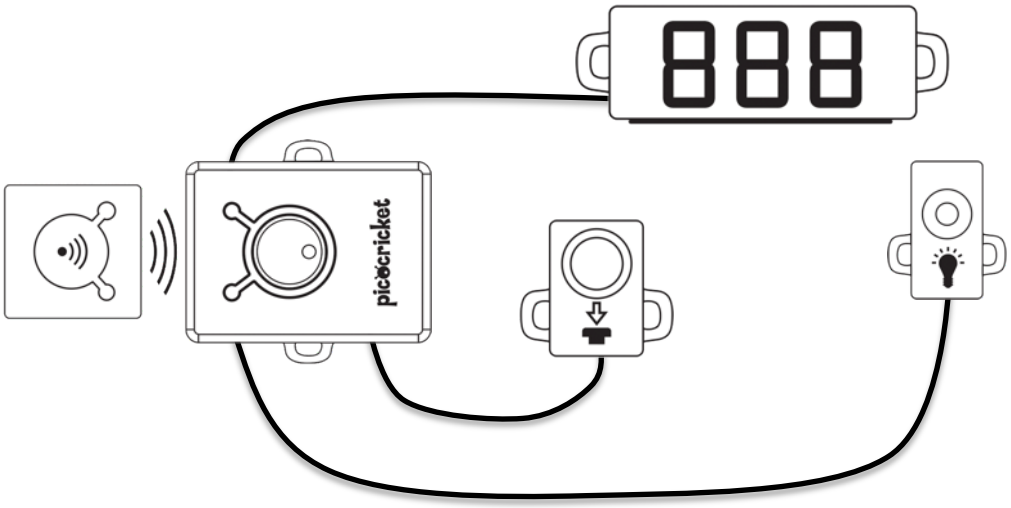
The goal of this experiment is to know how long are the reflexes of somebody. With this test we will measure how much time a person needs to press the button after seeing the LED turning on.

Algorithm
The functioning of this algorithm is to repeat X times (5 in our experiment) the getting started a timer when the light is turned on and stopping it when the person presses the button. Once finished (after 5 times) the algorithm calculate the average time.

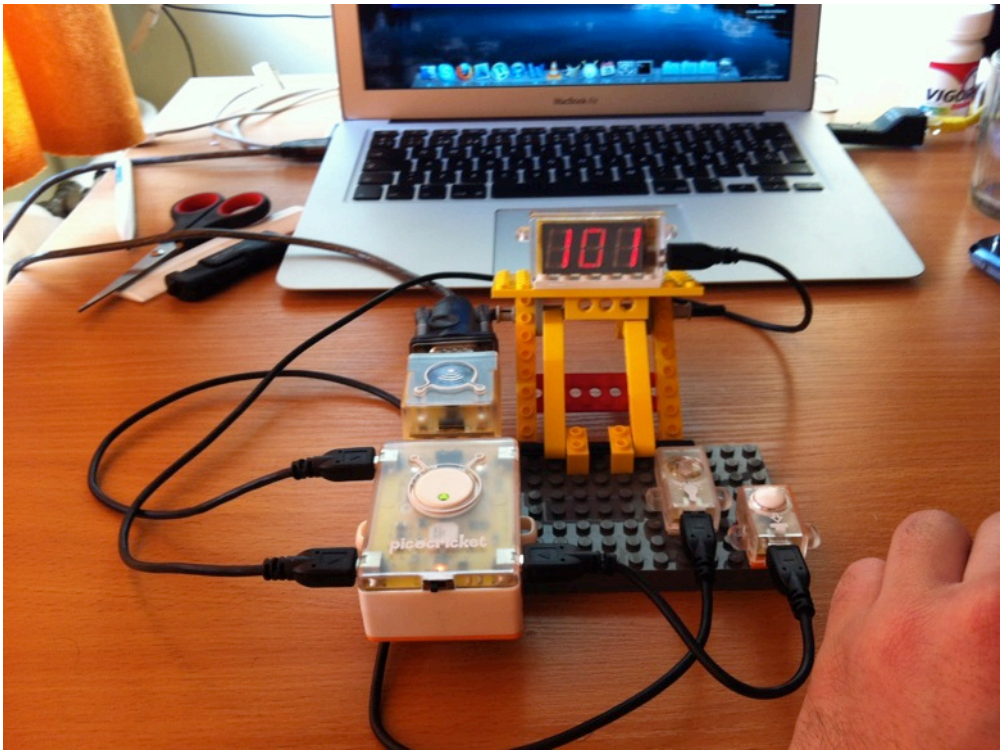


Schema

We only need the display, the button and the light to conduct this experiment.



After having everything ready we had:



After a set of 5 samples we could get the following results for our sight reflex speed:

#1	#2	#3	#4	#5	Average
57ms	20ms	24ms	25ms	16ms	28ms

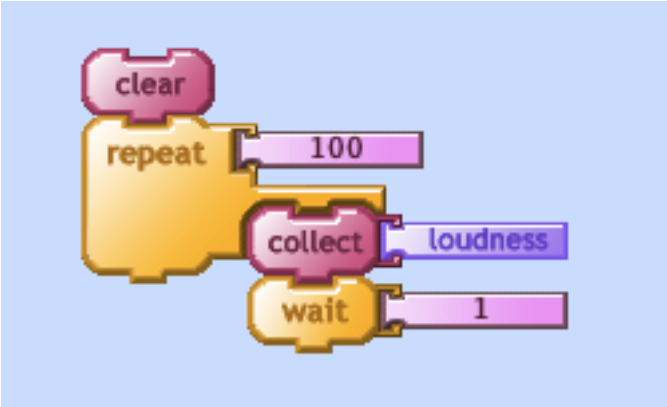
As a result the average time of our reflexes is 28 milliseconds, despite we would need to repeat it with hundreds or thousands samples to get more precision.

Sound detector experiments

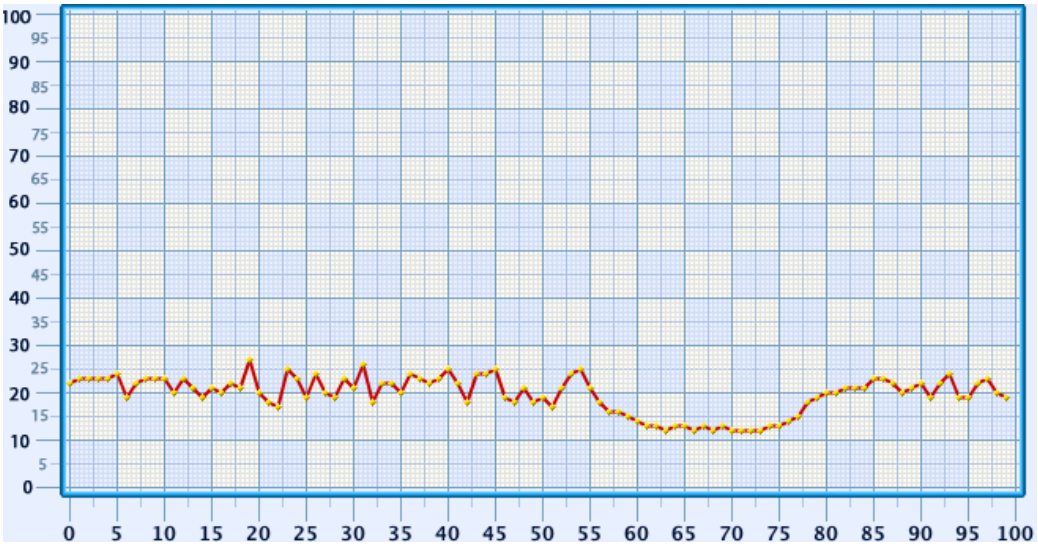
The sound detector can measure the volume of incoming sounds. The volume of the sound is given in a value from 0 to 100. Zero is no sound and 100 means a loud noise.

1) Getting the intensity of sound

It consists on getting the graph of the volume of an incoming sound heard by sound detector during 10 seconds (measuring 10 times each second). The algorithm is:



As a result, the obtained graph for a certain song is:

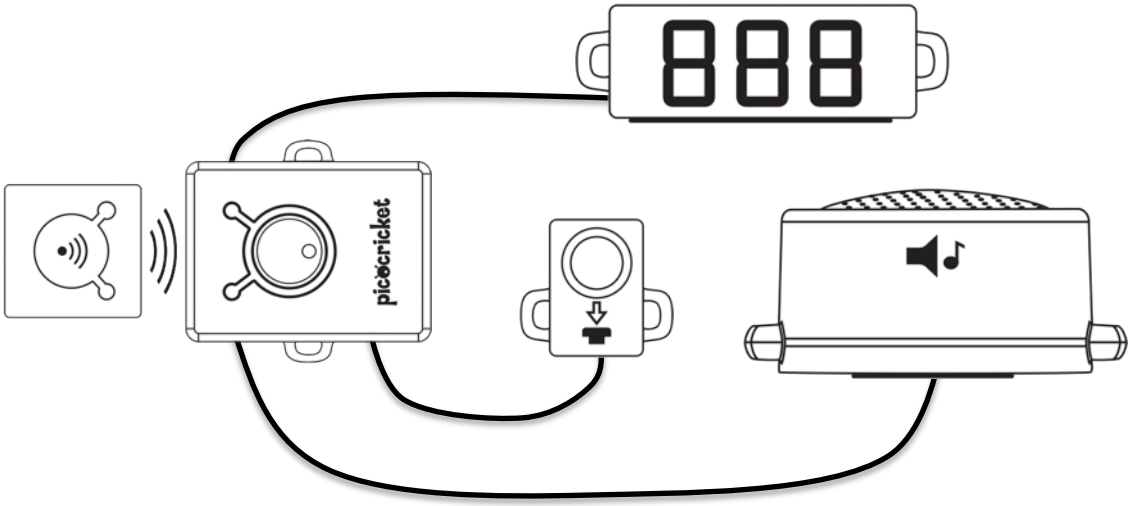


There is also the possibility to get the output as a list of measurement for each sample taken.

2) Reflex experiment with sound

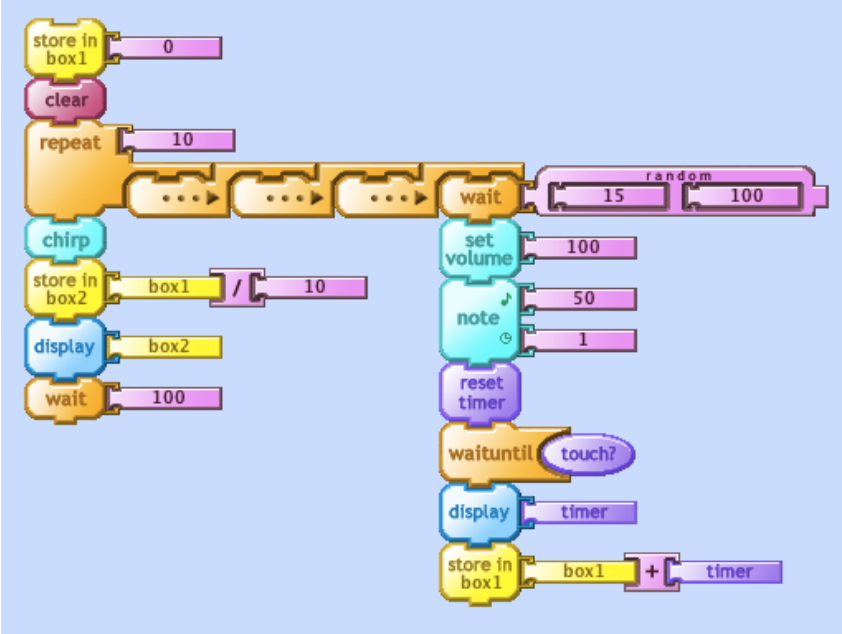
The aim of this experiment is to check how quick is somebody from a sound is played to he press the button. The following experiment has some similitudes with the same experiment with light, but instead of using the LED as signal, we use a sound signal.

Schema

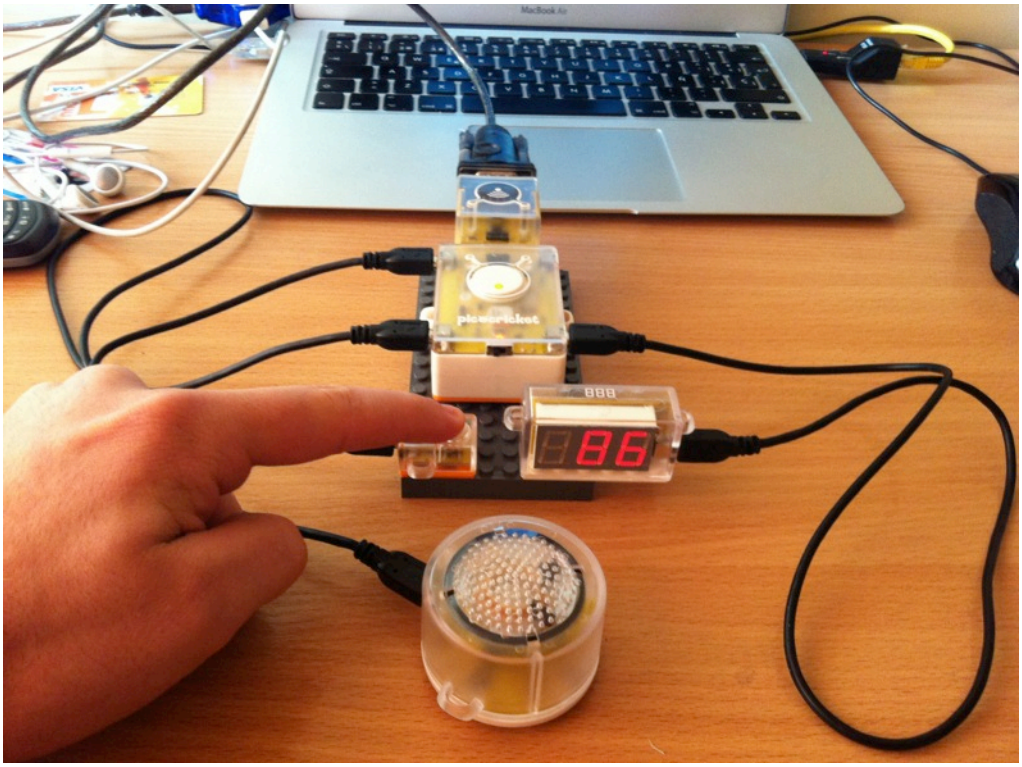


Algorithm

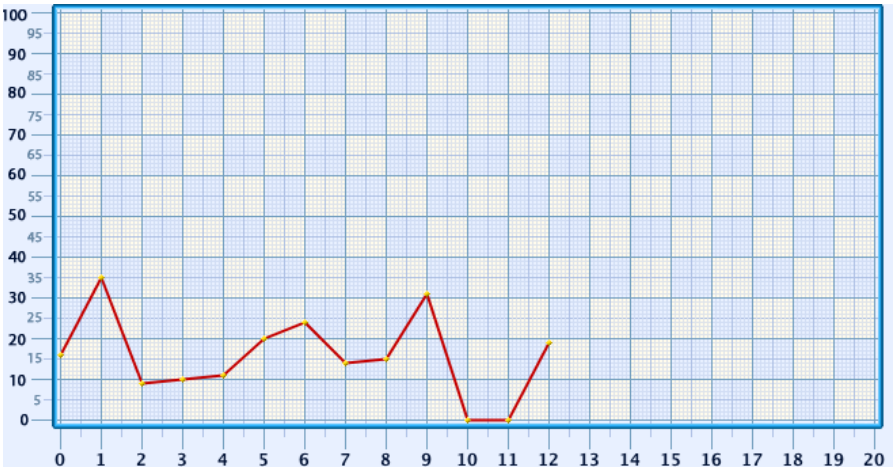
As we did with the reflex experiment using a LED, now the algorithm is quite similar. We need a 10 repetition's loop and each time we **wait** unknown time (from 1,5 to 10 seconds), then we set a sound and its volume, afterwards we turn the timer on and we just **wait until** the button is pressed. When it occur we just show the timer on the display and we add this time to the memory block **box1.** After 10 times, we just need to **chirp** to let the user know it has finished. The next step is to calculate the average time, so we save into **box2** the sum divided by 10, which is the average time.



The experiment once conducted:



We can also insert a **collect** operator to have a log of the different times we get.



After plotting the 10 measurements, the algorithm plotted two 0 values to separate the average plot at the end.

The obtained values were:

#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Average
16ms	39ms	9ms	10ms	11ms	20ms	24ms	14ms	15ms	31ms	19ms

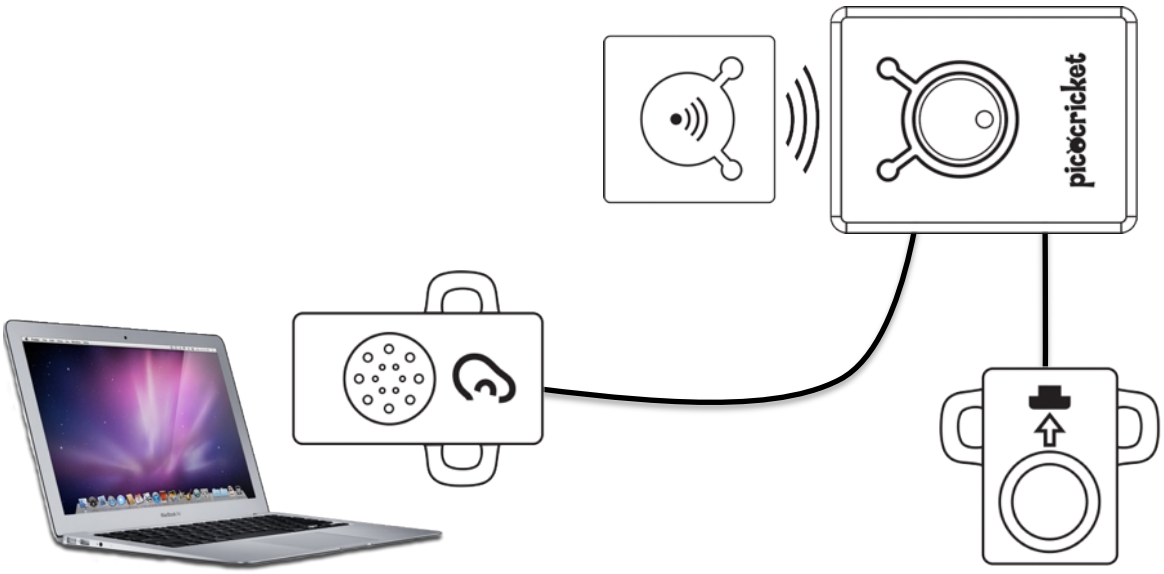
Here we got the time of 10 samples, which is more precise than the experiment we did with light. The average time is not precise enough, but is more reliable than the average time we experimented with light.

3) Volume of frequency

The aim of this experiment is to check if all range of sound frequency has the same volume intensity. When we did the same experiment with light frequency we could verify not all light frequency emits at the same intensity. Consequently we want to test the same fact but with sound frequency.

We just need the microphone and a sound source to play all frequencies of sound. As we did in the same experiment with light, we will play the range of frequency from 20Hz to 20000Hz from a sound file from the computer. The sound track is 20 seconds long.

Schema

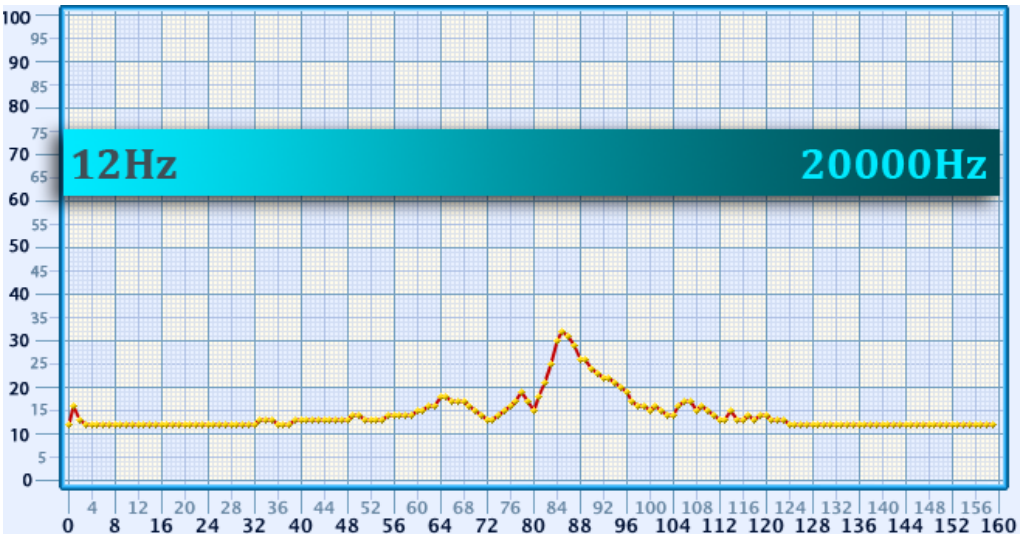


Algorithm

We need to play the sound track and immediately press the button to start registering the volume intensity. Once we press the button, the main thread must enter into the registering loop. The loop will last 16 seconds while writing intensity data every 0'1 second as showed in the following picture:

Results

After having conducted the experiment the drawn plot was:



In the left side we have the intensity values and in the bottom of the plot we have the different frequency values starting from 12Hz until 20000Hz (20kHz).

Quick conclusion

We can summarize as well as different light frequency has different intensity values, we have the same situation with sound instead of light.

Reflexes conclusion

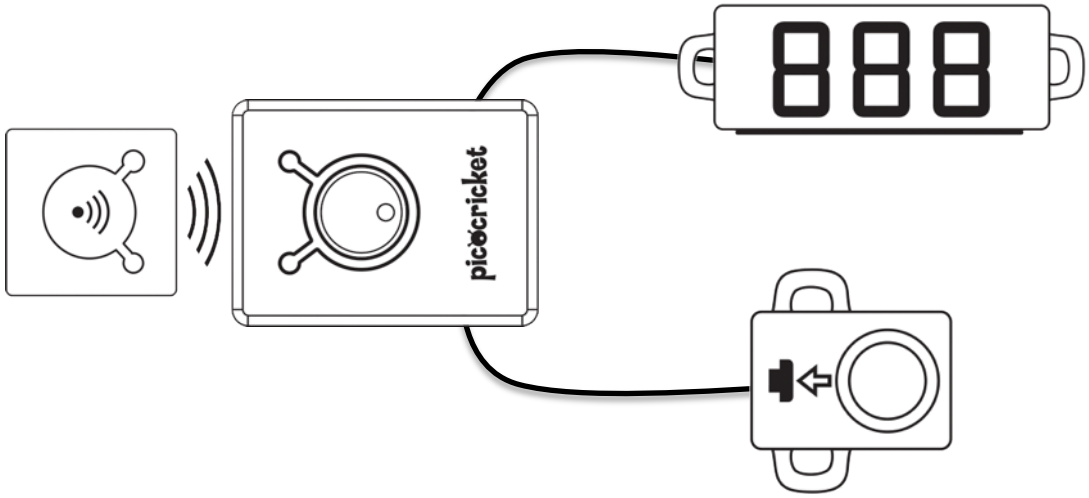
The main conclusion we get is we have faster reflexes using our ears than using eyes. That's why television has 24Hz, which is enough to make us believe that the image is moving. If the time between images were less, we couldn't notice a fluent movement. On the other hand, if this time was shorter we still could see the same movement on the screen but with a lot of extra information to process, it would be useless information because our eyes couldn't see it, so we would move more information than needed. It is known that the proper time between photograms is the time used on 24Hz video, not more than needed neither less than makes the video unwatchable and really slow.

As we have slower reflexes using our eyes instead of our hears, codifying video information is slower because we need to codify more images per second, and an image contain more information than a sound track.

4) Human hearing range

We humans are able just to hear between 2 specific frequency sounds. In the lower frequency it has been demonstrated that we can hear from 12Hz until 20000Hz as the highest frequency. But that's not the same for everybody because it depends on the age of who's listening, or even it depends on the physical state of the person in that moment.
The experiment is simple, a sound it's played at the same time the experiment is running and when the person can start hearing some sound he press the button. Then an approximation of the frequency he can hear is shown in the diplay.

Schema



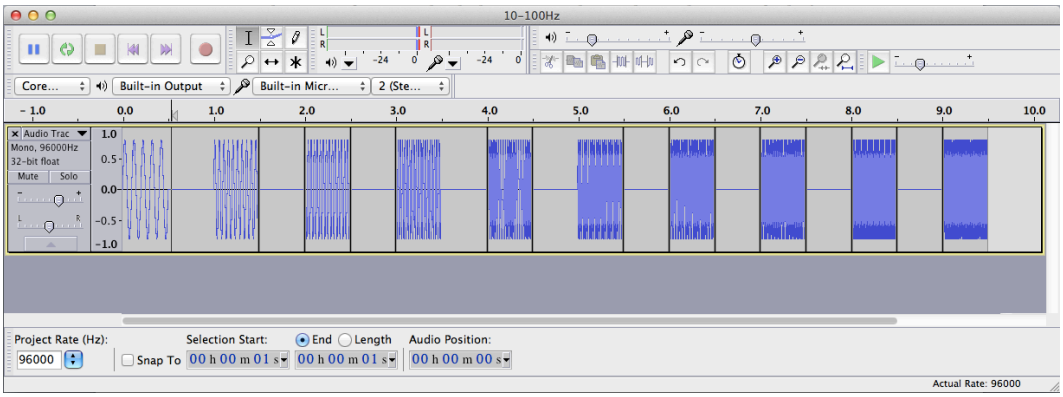
The real assembly for the experience is:

Experiments

The aim of the following test is to determinate what is the lowest frequency a person can hear and also what is the highest.

Determination of lowest audible frequency

For this experiment it's needed an external piece of sound with an increasing frequency from 0Hz to 100Hz. This sound has been made using Audacity which allows us to create custom sounds with a determinate frequency.



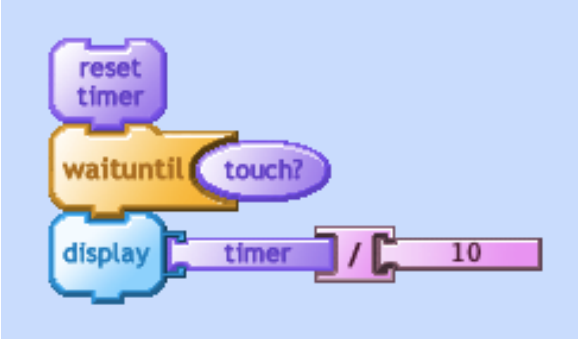
The custom sound is split every second and each second contains 500ms of the frequency we need and 500ms of silence to be able to notice the change of frequency. In that case we could summarize this track as follow:

#	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Second	0 - 1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10
f	10Hz	20Hz	30Hz	40Hz	50Hz	60Hz	70Hz	80Hz	90Hz	100Hz

We can find a relation between the time in the sound interval and the frequency which is been playing.

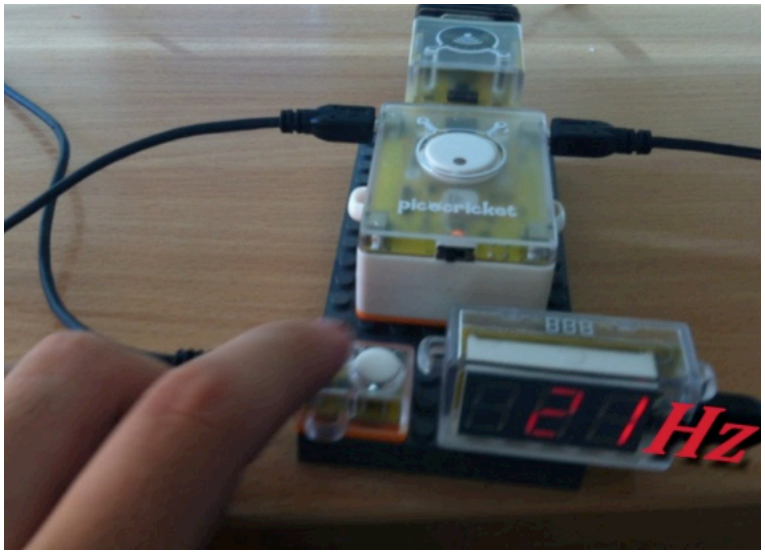
I used the same construction to conduct both experiments because I just need to change the algorithm to conduct the experiment of low frequency and a small variation to conduct the higher frequency one.

Algorithm to find out the minimum hearing frequency



This algorithm resets the timer at the beginning of the execution then it waits for the button being pressed and immediately it displays an approximate frequency. As a parameter we just have the time from the beginning but it's enough to know the frequency because as we see in the table we just need to divide the timer by 10.
The most important to get better precision is to play the sound file at the same time we load the algorithm in the PicoCricket system.

Here a test:



The result of that experiment was that the minimum frequency I can hear is 21Hz this is just an approximation because this experiment is too complex to get precision.

We can get some more accuracy if we approximate it using the population mean of several samples. I took 5 samples (n=5) with the same experiment, but the more samples you get, the more accuracy you have.

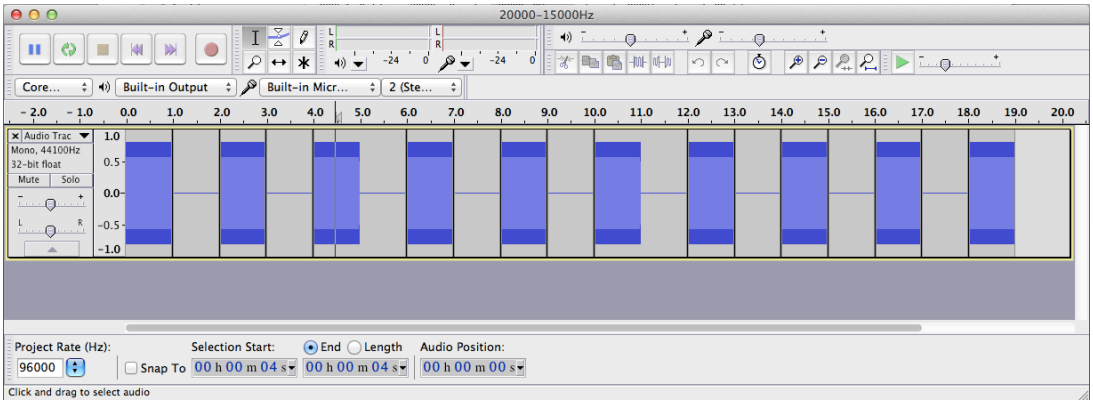
Here the results:

X_1	X_2	X_3	X_4	X_5	$\overline{X}$
19Hz	19Hz	18Hz	15Hz	21Hz	18,4Hz

We can consider that 18,4Hz is a more accurate value for the minimum hearing frequency. If we would do the experiment thousand times instead of 5 repetitions, we could have even a more accurate value.

Algorithm to find out the maximum hearing frequency

In that case the experiment is similar to the other one, but now we have higher frequencies, which will have to multiply by 1000 because they are in KHz. The sound interval we will use is the following:



It goes from 20000 decreasing to 15000. The algorithm is the same but we just need make a little change to the table we had before:

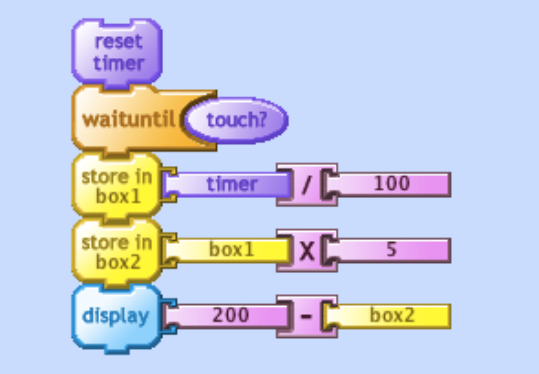
#	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Second	0 - 1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10
f	20MHz	19,5MHz	18MHz	18,5MHz	18MHz	17,5MHz	17MHz	16,5MHz	16MHz	15,5MHz

The absolute maximum is 20MHz therefore it is our reference. As the time goes increasing, we will go decreasing starting with our reference point. The value we will subtract will be the difference in MHz between the points we have tested. The formula I used to calculate the frequency of a test in function of the number of the test is:

$$f = 20MHz - 5(\frac{timer}{100})$$

From timer/100 we get the # of test, and as we need to get it from 500 to 500 we just need to multiply by 5. Finally we just subtract this result to 200 to get the result in MHz using the last digit as if it was decimal.

Algorithm



The execution starts resetting the timer and afterwards waiting until somebody press the button, which means the person can hear the sound. Then we save the value of timer in seconds, then we multiply that value by 5 and finally we subtract it to 200 to display the final result in MHz

The display is showing the value “200” but we must understand that it means “20.0”. It will be the same for all values shown.

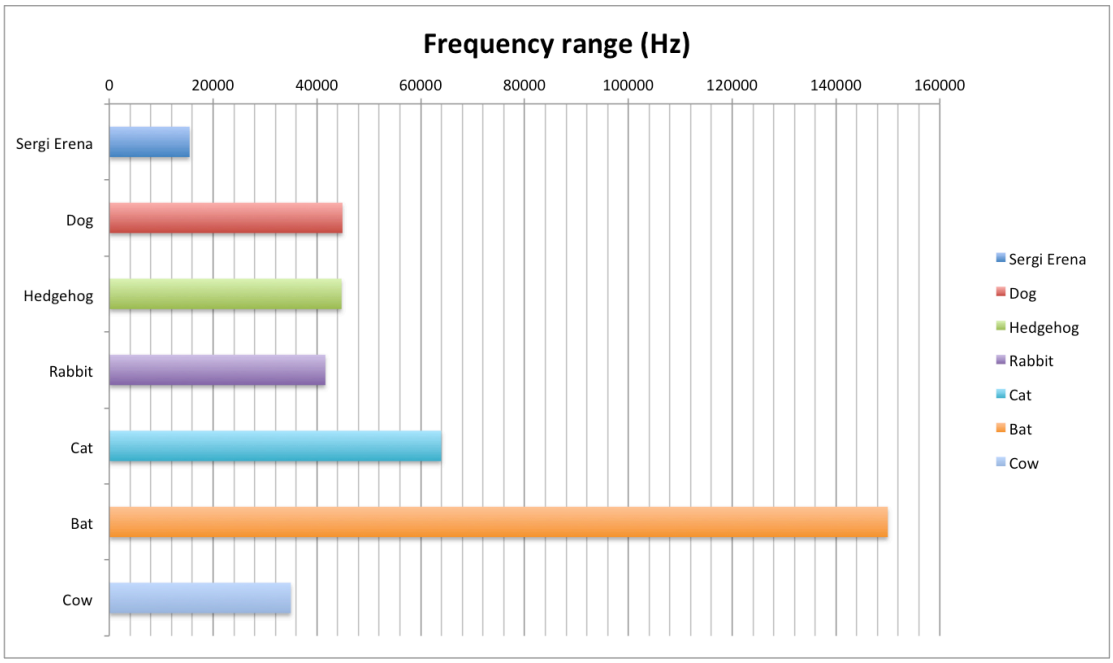
This is just the result of one single test and it’s difficult to determinate the exact frequency value because we should need to repeat the experiment more than once. We need to keep all results we’ve got in a table to get an average value using the statistics.

After having repeated the same experiment 5 times we got the following results:

X_1	X_2	X_3	X_4	X_5	$\overline{X}$
14,5MHz	15,5MHz	16MHz	15,5MHz	16MHz	15,5MHz

So then we can conclude that the maximum frequency I can hear is approximately 15,5MHz.

Knowing the minimum and the maximum I can hear, we can finish up saying I can hear from 18,4Hz to 15500Hz, which means 15500-18,5= 15481,5 frequency range.
Compared with other animals, human range is really reduced, so we are missing thousand sounds we cannot hear. There's the comparison



Observation

Speakers have different properties that make every single test different from the previous realizations. Probably the result of that experiment with different speakers could have different results due the technical features of speakers. We can summarize them as:

- Frequency response: This is the spectrum range the speaker is able to reproduce. The better the speaker is, the widely range

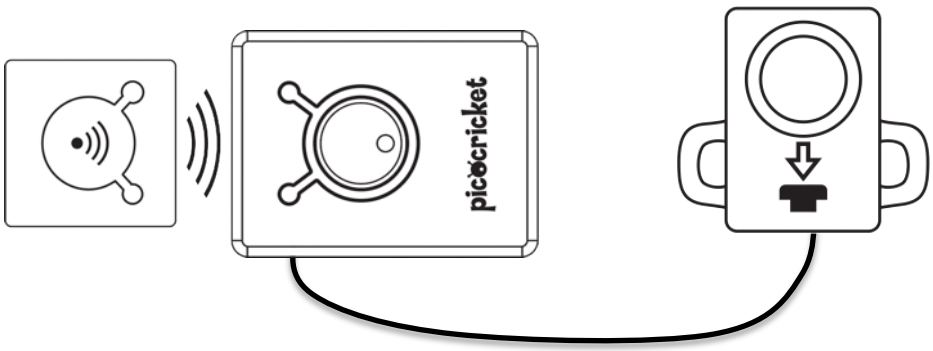
it can play. With a bad quality speaker we are not available to reproduce all the human hearing range.

- Features linked with power: Nominal power, average power, PMPO (Peak Music Power Output)…
- Impedance: This is the electric resistance produced by the speaker when it is receiving a 1kHz signal.
- Sensibility: This value is the relation between the electric input signal and the sound generated by the speaker.
- Efficiency: Is the sensibility expressed in %.
- Distortion: This is the smallest sound defect the speaker produce. They are generally due when very low frequency sounds are being played.
- Directivity: This is the feature which spreads the sound out of the speaker in all directions.

As we have different feature differences between speakers (actually there aren't two equal speakers) the conducting of this experiment was difficult.

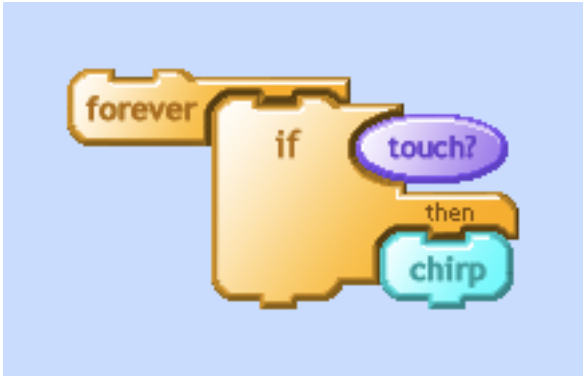
Touch detector experiments

The touch detector is a simply button that can detect if it is pressed or isn't. The incoming data for this detector is boolean data being not pressed → 0 and pressed → 1. To carry out all those experiments the hardware we need is only the touch sensor. So the schema is the following:



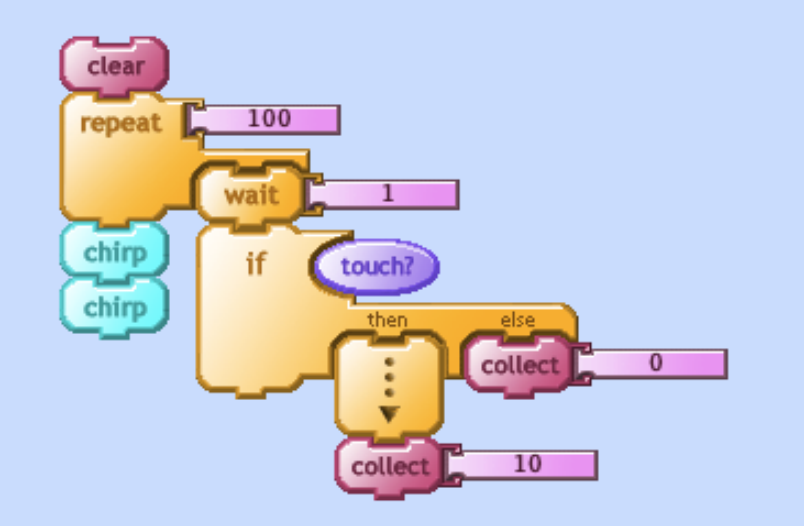
1) Chirping when pressed

The experiment consists on a simple detector to know when the button is pressed. It is the same performance as a house bell, if somebody presses it, the bell sound.

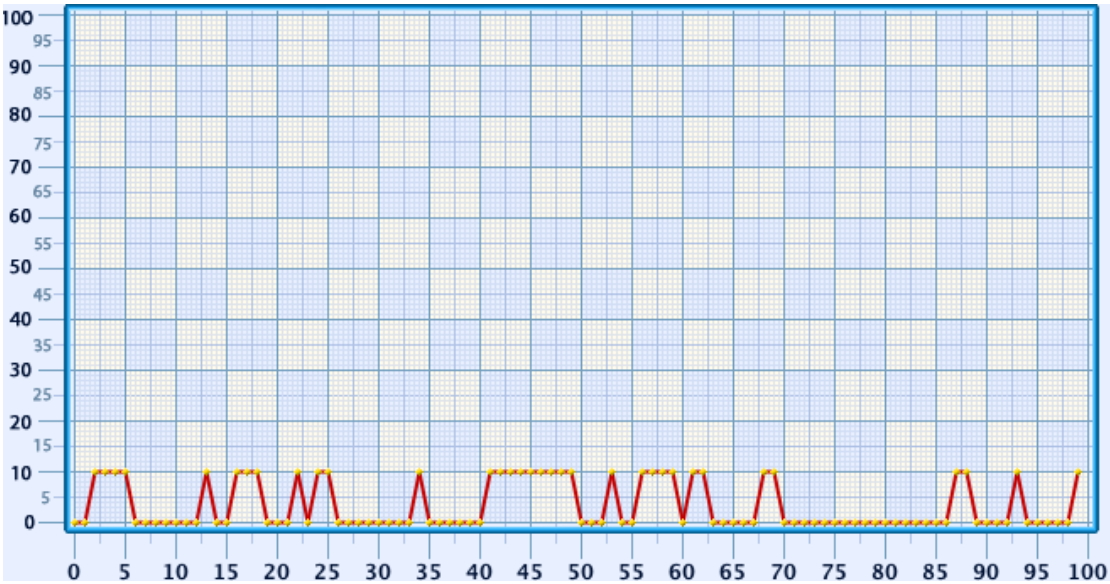


2) Plotting touches throw time

This experiment consist on keep a register of touches. To make easier the viewing of this touches history we will keep the information in a graphic, being 0 if the button is not pressed and 10 if it is. The experiment takes 10 seconds to draw the touches of the button. The algorithm is the following:



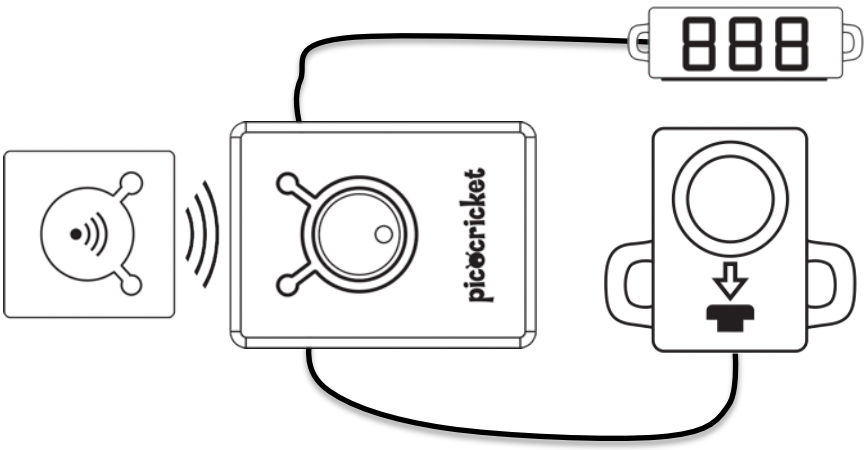
After running it with a random pressing combination the resulting plot is the following:



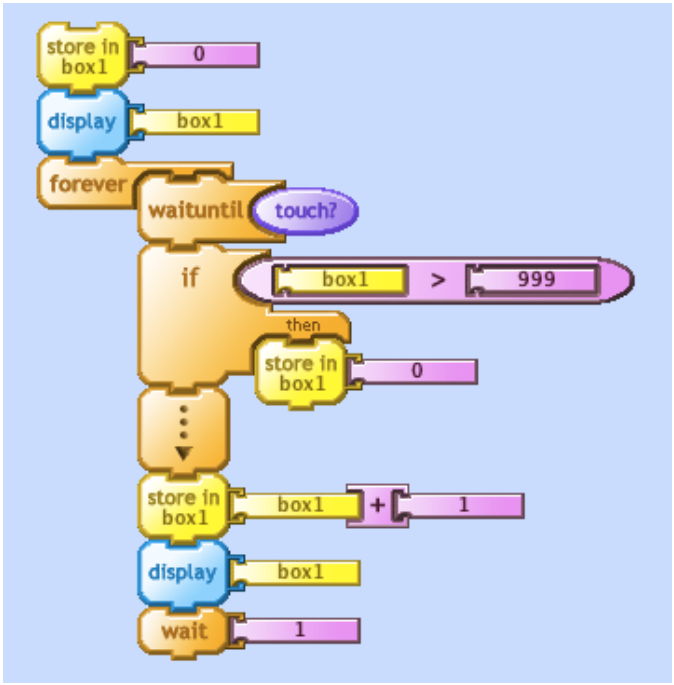
We can see there is as a boolean graph so we can conclude the button can be used as a boolean input.

3) Touch counter

The aim of this experiment is to have a register of how many times we pressed the button. We will need a place to store the number of times pressed, so we will use a memory register to increase each time we press. We need the display to show the number of times pressed. The schema will be:



The algorithm will be the following one:

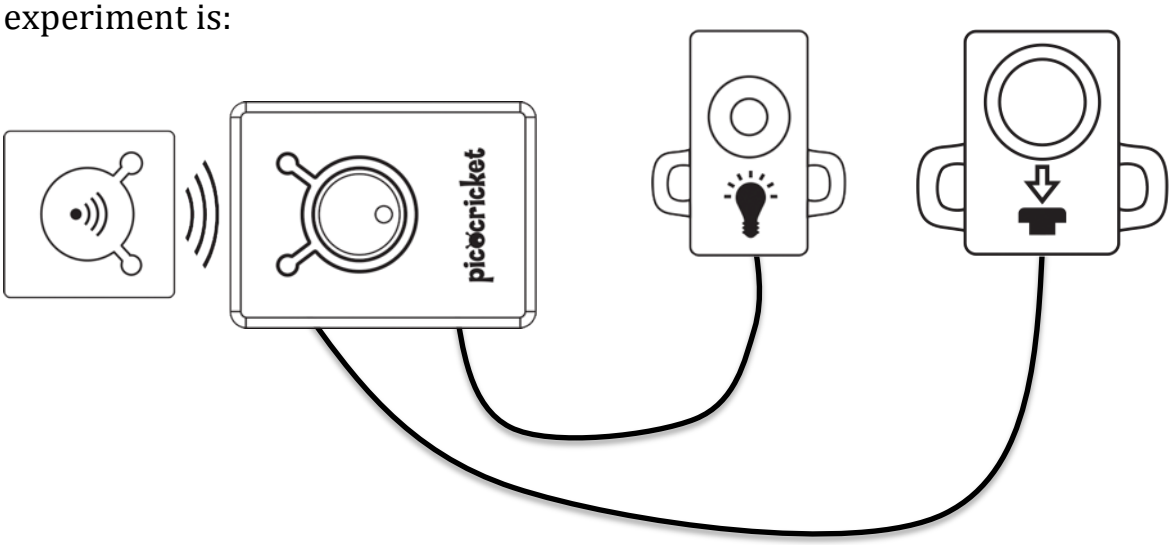


As the display can only show from 0 to 999, we have an overflow control using a conditional instruction to restart the counter in case it reach 999.

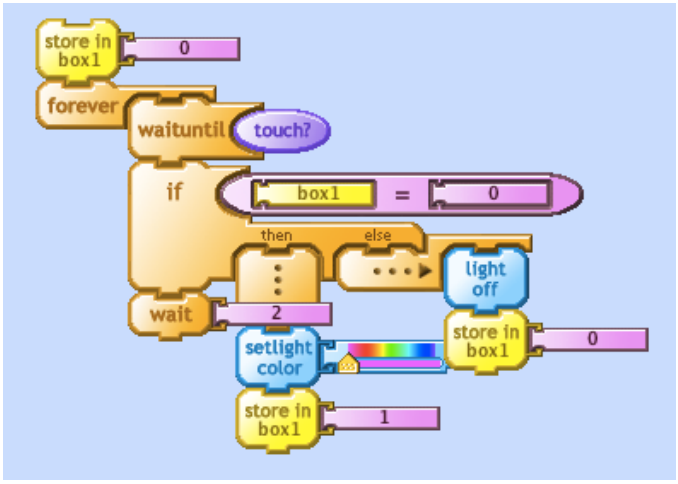
4) Touch sensor as a switch

Another use for the touch sensor is to use it as a switch. For example it can be useful to turn on/off a light with the same button. We need to know if the light is turned on or not, so we will have a memory register to check it with a boolean value (0 or 1). After switching we

have to update the boolean value manually. The schema for this experiment is:



And the algorithm for the experiment is the following:

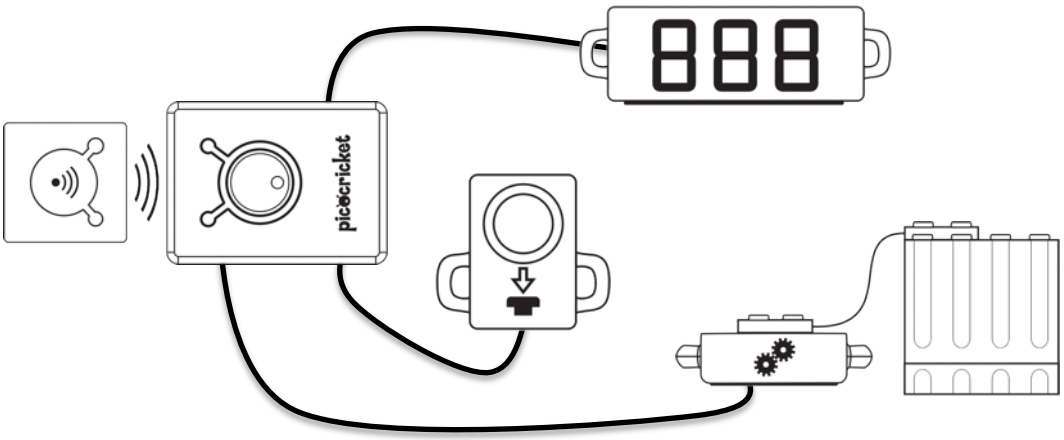


After wait until touch, we check if the light is turned on or off, and in case it is off, we turn it on and we save the change into the memory, otherwise we do the contrary if the light is on.

5) Reflex experiment using touch sensibility

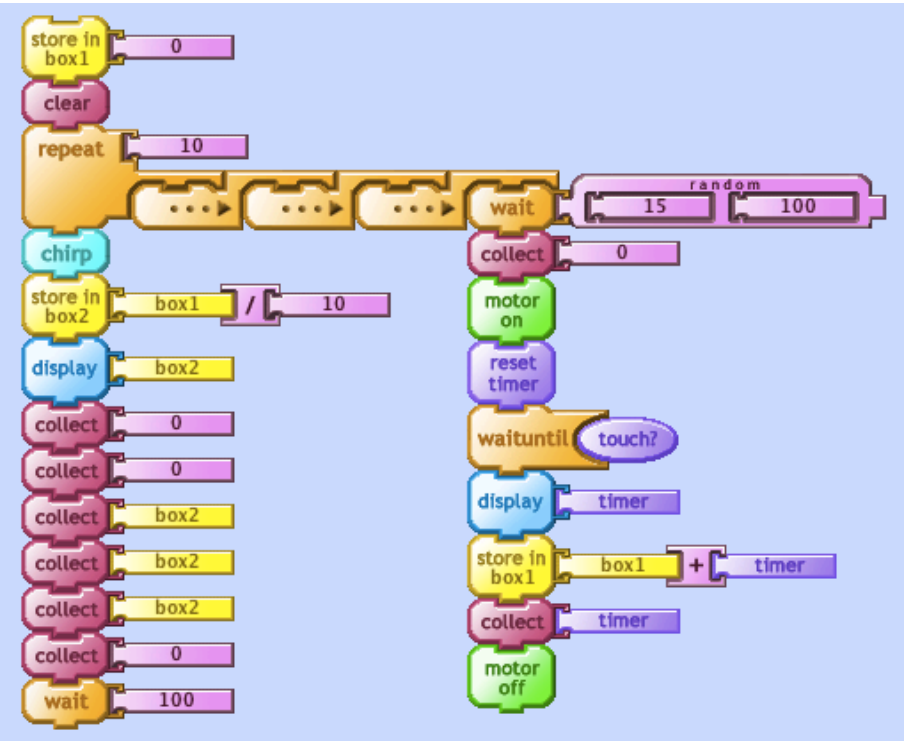
After testing reflexes using light and sound signals it's time to try the touch as a signal. The signal for this experiment is a mechanic signal and we will use the motor included in the PicoCricket system.

Schema



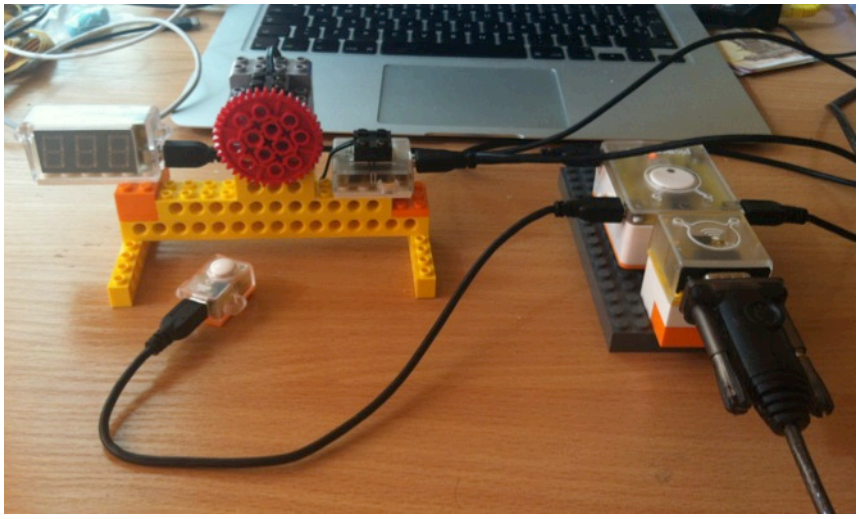
Algorithm

The algorithm we will use to conduct this experiment is just a loop that turns on the motor and the timer at the same time, and then it wait until the person press the button. After this action the display shows the time the person has needed between he noticed the movement and he pressed the button. Then we add the result in a memory block in order to calculate the average reflex time later.



This sequence is done 10 times in order to get more data therefore we get more precision. At the same time, the algorithm also write a plot with the time of each repetition, and after 10 times, the average time is written in the end of the plot separately from the other results.

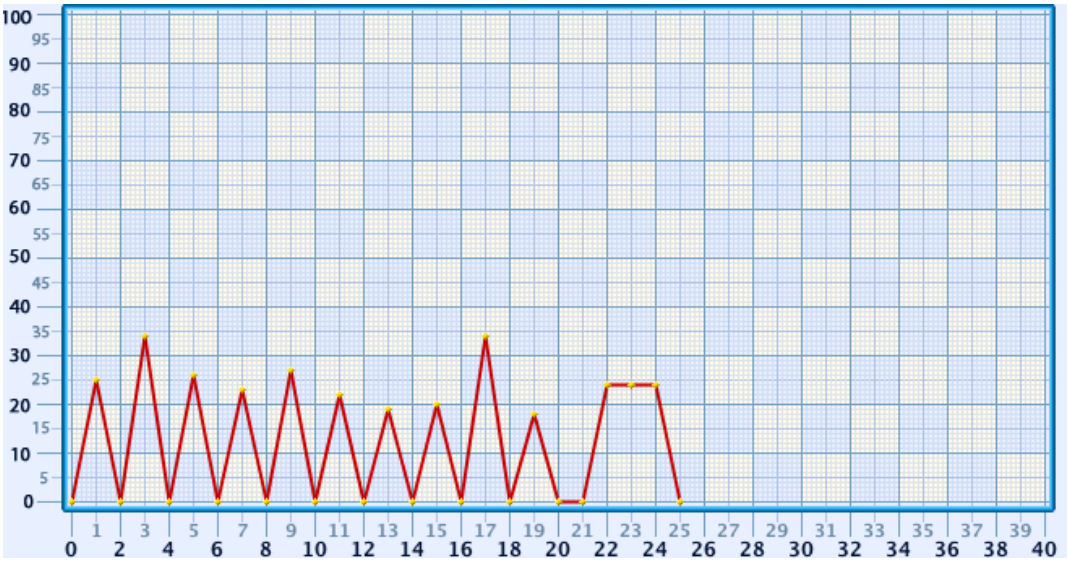
The experiment once conducted was



In order to detect the movement, the person has to put his hand over the red wheel.

Results

After doing the experiment with myself, the results I could get were:



On the left side of the plot there is the time in milliseconds and in the bottom there is the number of repetition. After 10 times, in the end there's the average time calculated. Here's the table with real data:

#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	AVG
25	34	26	23	27	22	19	20	34	18	**24**

Conclusion

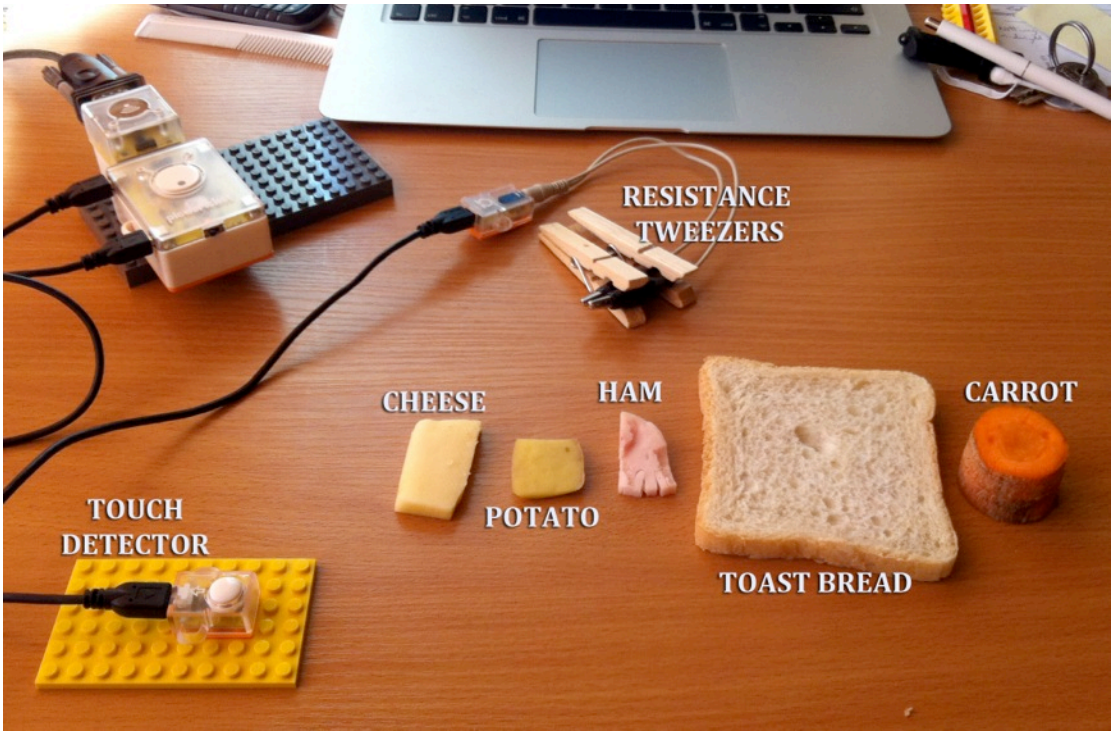
Our touch reflexes are quicker than our light reflexes and a bit slower than sound reflexes. This conclusion is not exactly because it depends on the situation the experiment was done. If the experiment is conducted in the morning, the results are different than if it's done in the afternoon or even in the middle of the night.

Resistance detector experiments

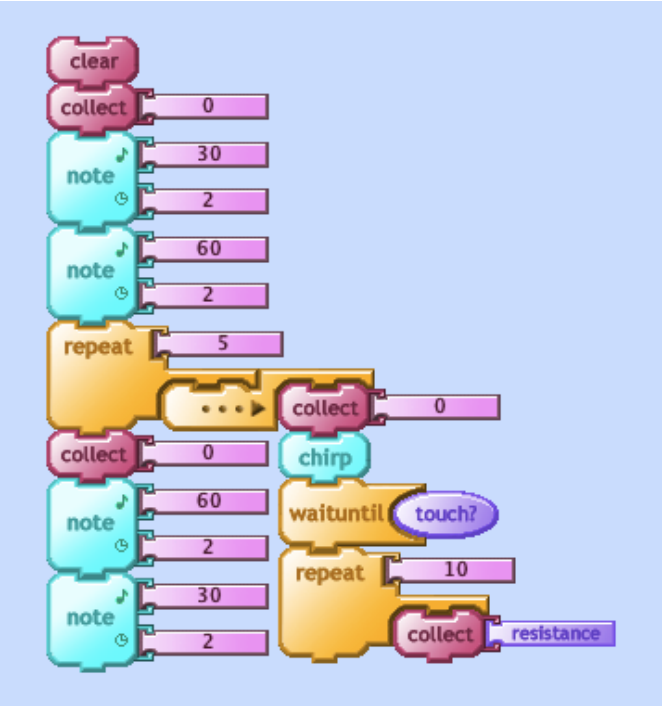
The last sensor I worked with was the electrical resistance sensor. We just have to connect the 2 tweezers to any kind of material. The PicoCricket System can measure the electrical resistance between the 2 tweezers. The resistance can get a value between 0 (low resistance) and 100 (high resistance). The PicoCricket allows you to delimitate the range of measurement because some times we need more precision.

1) Getting the electrical resistance of some food

A good experiment to compare different material resistance is measuring the resistance of 5 different pieces of food. To conduct the experiment I used cheese, potato, ham, toast bread and carrot.

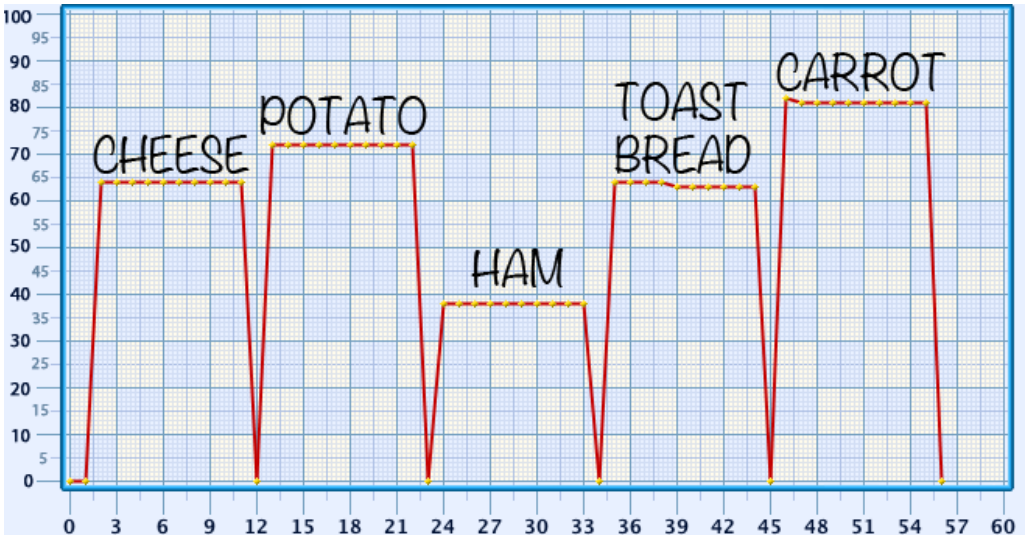


The experiment consist on put the tweezers in contact with the material to test and afterwards pressing the button to get the material measurement. The algorithm is:



In each loop the sequence the algorithm first plot a 0, then it makes a sound signal with **chirp,** then it keeps waiting until the button is pressed, and finally it enter to another loop that keeps the resistance value every 10ms. This value is kept 10 times in order to be more precise and clear in the feedback plot.

When the button is pressed, the algorithm keeps the resistance value in the graph storing memory to draw the plot. Once the experiment was done, the output results were:



To have a clearer plot, for each material we plotted the data 10 times. In the left side of the graph, there is the resistance in ohms, therefore we can sum up with the following table:

Material	Resistance
Cheese	64 Ω
Potato	72 Ω
Ham	38 Ω
Toast bread	63 Ω
Carrot	81 Ω

Conclusions

Now it's time to write my feelings after having the experience of working with that marvellous system. Before having done this project I never had heard anything about this system, but I am widely surprised and amazed how something so difficult and sometimes surrealist how electronics is, can be offered to users so easily.
Thanks to my previous knowledge about programming with microcontrollers, learning and putting in practise everything was easier than I expected at the beginning. Some experiments were quite difficult due they would need more accuracy to get better results, but considering that PicoCricket system is not a professional set of tools, I am really satisfied of.
At the begging everything was unknown for me, so I had to read a lot of documentation to get used to the system, but once I knew how everything worked, I realized about how easy it is. The easiest experiments and more interesting were the ones with light, because they always worked properly. In contrast I had more problems with sound experiments, because sometimes I had to generate sounds with a specific frequency, and I had to use external software because the PicoCricket system doesn't have that feature.
I really enjoyed conducting the electric resistance experiment with different aliments because it worked perfectly, without problems neither variations in the final result, and furthermore I could eat some food.
The most difficult sensor to make experiments is the touch sensor, because its simplicity, so I had to be original and think on original experiments using a simple button. After all I'm proud of what I did with it.

Moreover I'm happy of having written this project in English, because I am very interested and motivated on my English improvement but I think I'm not good enough. At the beginning I had the option of doing this project in Catalan, in Spanish or in English, and following my interests I choose do it in English. And now that I'm writing the lasts lines of it, I'm greatly proud of myself because I could achieve it.